*\*

***SUBMITTED TO:->******SUBMITTED BY:->***

*ER.VIVEK MAHAJAN PIYUSH MEHAN (4106)*

*PRINCEPAL SINGH (6859)*

*RAJBIR SINGH (4112)*

**ACKNOWLEDGEMENT**

*We owe a great many thanks to a great many people who helped and supported us during this project. We deepest thanks to Mrs. SHIVANI MAHAJAN & Mr. VIVEK MAHAJAN  the Guide of the project for guiding and correcting various documents of us with attention and care. They has taken pain to go through the project and make necessary correction as and when needed. We deep sense of gratitude to ICLT for support and guidance. Thanks and appreciation to the helpful people at Satyam polytechnic college for their support.*

*About Project*

*Employees are the backbone of any company and their management is key in determining organizational efficiency and risk mitigation. Employee Database Management is an important employee management practice that facilitates employee record maintenance, correct payouts to employees, statutory due calculation and salary expenses. With businesses having a growing workforce across multiple locations, a centralized and efficiently managed employee database system becomes critical in ensuring easy and correct information access. CRP's Employee Database Management Services help companies manage the entire employee record management cycle in an efficient and cost-effective way.*

*CRP has dedicated and trained manpower and also uses a continuous Risk Matrix of physical check of employees in office premises, checking of the records and IT led internal controls to ensure cost savings and a risk free database management system for its clients. Most Employee Database Management systems are weak on implementing correct and on time record updation. Such lapses can lead to losses by way of salary, statutory dues and other expenses paid to employees who are no more with the organization. Also, it can lead to incorrect MIS reports and wrong manpower analysis. CRP's Database Management Services handle end-to-end or select Database Management jobs for companies to help them in efficient employee management and increase HR and Administration bandwidth. Talent crunch in a global work environment have made Employee Background Check & Pre Employment Screening indispensable to building a professionally ethical workforce. The very basis of productivity and growth, an employee is also the custodian of a company's reputation and goodwill.*

*On the other side, an employee can be a major people risk to a company. Therefore, hiring right and determining the authenticity of information provided by a prospective employee is a necessity for any business. CRP understands the criticality of hiring process in reducing people risk for a company. Through our Employee Background Check & Pre Employment Screening services we provide companies comprehensive solutions in the recruiting domain that are customized, cost-effective, and value-adding.*

*Through its near two decades' experience, CRP has built an Indian and MNC client list of marquee names across IT & ITES, BFSI, Manufacturing, Retail and Services. With our seasoned team of professionals, support staff and all-India network of branches, clients are assured of professional expertise and excellent service delivery. Employee Database Management is an important employee management practice that facilitates employee record maintenance, correct payouts to employees, statutory due calculation and salary expenses. With businesses having a growing workforce across multiple locations, a centralized and efficiently managed employee database system becomes critical in ensuring easy and correct information access. CRP's Employee Database Management Services help companies manage the entire employee record management cycle in an efficient and cost-effective way.*

*CRP has dedicated and trained manpower and also uses a continuous Risk Matrix of physical check of employees in office premises, checking of the records and IT led internal controls to ensure cost savings and a risk free database management system for its clients.*

*KEY RESPONSIBILITIES:*

* *Manages the employment process which entails the recruitment, recommendation for selection and placement of all support and some professional employees.   Activities include drafting and placing ads; working with employment agencies; skills testing of applicants; interviewing, reference checking and salary negotiations.*
* *Manages job offer process for approved hiring with final approval from COO.*
* *Manages human resource files for staff below senior management level, establishes/maintains an HRIS system, resume database*
* *Identifies and maintains relationships with recruitment agencies*
* *Conducts orientation for new employees and ensures that all paperwork is completed in an accurate and timely manner*
* *Assists staff on a variety of personnel related matters including sensitive issues, job reclassification, terminations, personality conflicts, potential areas of discrimination and policy interpretation.*
* *Monitors and provides input on personnel policies, procedures and practices to ensure that they meeting both the intent and spirit for which they were created.*
* *Monitors documentation for personnel files*
* *Drafts position descriptions and employee communications; working closely with COO*
* *Works with management in interpreting HR policies and procedures; updates TIP’s Employee Handbook as needed.*
* *Manages the performance appraisal process; assists in establishing guidelines for salary increases.*
* *Assists in maintaining TIP’s established salary structure.*
* *Administers the employee benefits programs which include health/major medical/dental/vision plans, life/ADD insurance, workers’ compensation and unemployment compensation.*
* *Assists management in identifying outside training for employees as deemed appropriate*
* *Completes other duties as assigned*

*SKILLS:*

* *Ability to maintain strict confidentiality - essential*
* *Strong recruiting experience*
* *Strong communication skills in both employee presentations and one on one conversations with employees regarding performance*
* *Excellent Microsoft Office 2007 experience*
* *Detail oriented, organized; ability to handle a fast-paced, rapidly-expanding environment where priorities change frequently*

*Must be self-motivated and able to anticipate needs and potential problems; self-starter. Due to the nature of computers and technology, the need for highly trained database administrators will always exist. Database administrators are needed in just about every sector, but their numbers are highest in the computer systems design and related services industry. The industry can expect to see more independent contractors, temporary employees, seasonal employees, and telecommuters.*

***Salary Trends*** *In 2009, database administrators earned an average salary of $67,250 per year. This figure represents a 3.8% increase over 2008 ($64,670). Database administrators in the 90th percentile can expect to earn around $103,100 per year, while 75th percentile database administrators can expect to earn $84,830 per year. Entry-level database administrators can expect to earn a starting salary of around $37,350 per year.****Degrees and Training Programs****Most employers prefer individuals with a bachelor’s degree or higher in computer science, information science, or management information systems. Although a bachelors degree or higher is preferred over, say, an associate’s degree, more and more employer’s are seeking out individuals with master’s degree in business administration (MBA) with a concentration in information systems. This growing trend is the result of the growing number of businesses that have or plan to move their business to the Internet.   
  
The majority of community colleges offer an associate’s degree in computer science or a related field. Independent technical institutions and proprietary schools also offer an associate’s degree in computer science or related information technology field. These occupation specific programs do not offer advanced training, but rather training that will help aspiring database administrators get a foot in the door of local businesses. Advanced training may be acquired through a 4-year degree program, followed by a master’s degree program. Just about every 4-year university or college offers a computer science and/or related information technology degree.****Coursework Required*** *In addition to core courses from the social sciences, life sciences, and humanities subject areas, aspiring database administrators can expect to take undergraduate classes such as software engineering, principles of programming languages, data structures, computer architecture, theory of computation and formal languages, operating systems, distributed systems, computer modeling, computer networks, compiler construction, and artificial intelligence.   
  
Master's degree seekers can expect to take software development, algorithms and structures, distributed systems, statistics, mathematical methods, logic, and numerical analysis, as well as specialized courses such as computational concepts in the sciences, computer architecture, bioinformatics, game development, and object-oriented programming. Completion of a computer science project or thesis is also required.*

*INTRODUCTION OF C++*

*OOPS CONCEPT*

*C++ is an object oriented programming language. It was developed by Bjarne Sroustrup at AT&T Bell Laboratories in USA developed it in the early 1980’s Stroustrup wanted to combine the best of both the languages and create a more powerful language that could support object oriented programming language.*

*The main problem associated with large programs written in procedural language is under evaluation if data. Data is neglected. There is no built in mechanism to ensure the security of data. The OOP approach is different. In OOP, the emphasis is on data not on procedures. In OOP, a class is the specification for such a data form.*

*In OOP, we bundle together the data and the functions that operate on the data into a single software unit called class. A class is the specification of the data entity. This data entity is called object. An object is an instantiation of a class. The relationship between a class and an object is the same as that of a built in data type and a variable of that type. A class is a data type and an object is an instance of that data type.*

*The most important and the most basic concept in OOP is the concept of the class. Define a class does not perform any memory allocation a class is like a blue print for a houses can be constructed. In a similar way you can create many objects of a given class. When an object is created, memory allocation takes place. OOP design methodology is different. Problems are no longer divided into functions. Instead it is divided into a collection of mutually interaction objects. Objects interact with one another through member functions.*

*Programming Language(C++)*

*C++ is the most widely used object-oriented language today. It is faster and moves powerful than java, another popular object-oriented language, which lacks certain features such as pointers and multiple inheritances.*

*Objects*

*Objects are the basic run-time entities in an object-oriented system. They may represent a person, a bank account, a table of data or any item that the program has to handle. They may also represent user-defined data such as vectors, time and lists. Programming problem is analyzed in terms of objects and the nature of communication between them. Program objects should be chosen such that they match closely with the real-world objects. Objects take up space in the memory and have an associated address like a record in Pascal, or a structure in C.*

*Classes*

*We just mentioned that objects contain data, and code to manipulate that data. The entire set of data and code of an object can be made a user-defined data type with the help of a class. In fact, objects are variable of the type Class. Once a class has been defined, we can create any number of objects belonging to that class. Each object is associated with the data of type class with they are crated. A class is thus a collection of objects of similar type. Classes are use-defined data types and behave like the built-in type of a programming language. The syntax used to crate an object is no different than the syntax used to create an integer object in C.*

*Data Abstraction And Encapsulation*

*The wrapping up of data and function into a single unit (called class) is known as encapsulation. Data encapsulation is the most striking feature of a class. The data is not accessible to the outside world, and those functions, which are wrapped in the class, can access it. These functions provide the interface between the object’s data and the program. This insulation of the data from direct access by the program is called data hiding or information hiding.*

*Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and defined as a list of abstract attributes such as size, weight and cost, and functions to operate on these attributes. They encapsulation all the essential properties of the object that are to be created. The attributes are sometimes called data members because they hold information. The functions that operate on these data are sometimes called methods or members functions.*

*Since the classes use the concept of data abstraction, they knew as Abstract Data Types (ADT).*

*Inheritance*

*Inheritance is the process by which objects of one class acquire the properties of objects of another class. It supports the concept of hierarchical classification. For example, the bird ‘robin’ is a part of the class ‘flying bird’, which is again a part of the class ‘bird’. The principle behind this sort of that each derived class shares common characteristics with the class from which it is derived. In OOP, the concept of inheritance provides the idea of reusability. This means that we can add additional features to an existing class without modifying it. The new class will have the combined features of both the classes. The real appeal and power of the inheritance mechanism is that it allows the programmer to reuse a class that is almost, but not exactly, what he wants, and to tailor the class in such a way that it does not introduce any undesirable side-effects into the rest of the classes.*

*Note that each sun class defines only those features that are unique to it. Without the use of classification, each class would have to explicitly include all of its features.*

*Polymorphism*

*Polymorphism is another important Object Oriented Programming concept. Polymorphism, a Greek term, and means the ability to take more than one form. An operation may exhibit different behaviors depends upon the types of addition. For two numbers, the operation will generate a sum. If the operands were strings, then the operation would produce a third string by concatenation. The process of making an operator to exhibit different behaviors in different instance is known as operator overloading.*

*A single function name can be used to handle different number and different types of arguments. This is something similar to a particular word having several different meanings depending on the context. Using a single function name to perform different types of tasks is tasks are known as function overloading.*

*Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface. This means that a general class of operations may be accessed in the same manner even though specific action in implementing inheritance.*

*Dynamic Binding*

*Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism reference depends on the dynamic type of that reference.*

*By inheritance, every object will have this procedure. Its algorithm is, however, unique to each object and so the draw procedure will be redefined in each class that defines the object. At run-time, the code matching the object under current reference will be called.*

*Message Passing*

*An object-oriented program consists of a set of object that communicates with each other. The process of programming in an object-oriented language, therefore, involves the following basic steps:*

* *Creating classes that defines objects and their behavior.*
* *Creating objects from class definitions, and*
* *Establishing communication among objects.*

*Objects communication with one another by sending and receiving information such the same way as people pass message to one another. The receiving information much the same way as people pass message to one another. The concept of message passing makes it easier to talk about building systems that directly model or simulate their real-world counterparts.*

*A message for an object is a request for execution of a procedure, and therefore will invoke a function (procedure) in the receiving object that generates the desired result. Message passing involves specifying the name of the object, the name of the function (message) and the information to be sent.*

*Objects have a life cycle. They can be created and destroyed. Communication with an object is feasible as long as it is alive.*

*Data Type*

*Variable names are arbitrary (with some compiler-defined maximum length, typically 32 characters). C uses the following standard variable types:*

*int -> integer variable*

*short -> short integer*

*long -> long integer*

*float -> single precision real (floating point) variable*

*double -> double precision real (floating point) variable*

*char -> character variable (single byte)*

*The compilers checks for consistency in the types of all variables used in any code. This feature is intended to prevent mistakes, in particular in mistyping variable names. Calculations done in the math library routines are usually done in double precision arithmetic (64 bits on most workstations). The actual number of bytes used in the internal storage of these data types depends on the machine being used.*

*The printf function can be instructed to print integers, floats and strings properly. The general syntax is*

*printf( "format", variables );*

*where "format" specifies the converstion specification and variables is a list of quantities to print. Some useful formats are*

*%.nd integer (optional n = number of columns; if 0, pad with zeroes)*

*%m.nf float or double (optional m = number of columns,*

*n = number of decimal places)*

*%ns string (optional n = number of columns)*

*%c character*

*\n \t to introduce new line or tab*

*Keywords*

*C++ keeps a small set of keywords for its own use. These keywords cannot be used as identifiers in the program — a common restriction with modern languages. Where users of Old C may be surprised is in the introduction of some new keywords; if those names were used as identifiers in previous programs, then the programs will have to be changed. It will be easy to spot, because it will provoke your compiler into telling you about invalid names for things. Here is the list of keywords used in Standard C; you will notice that none of them use upper-case letters.*

|  |  |  |  |
| --- | --- | --- | --- |
| *Auto* | *double* | *int* | *struct* |
| *Break* | *else* | *long* | *switch* |
| *Case* | *enum* | *register* | *typedef* |
| *Char* | *extern* | *return* | *union* |
| *Const* | *float* | *short* | *unsigned* |
| *continue* | *for* | *signed* | *void* |
| *default* | *goto* | *sizeof* | *volatile* |
| *Do* | *if* | *static* | *while* |

*Pointer :->*

*The C language allows the programmer to ``peek and poke'' directly into memory locations. This gives great flexibility and power to the language, but it also one of the great hurdles that the beginner must overcome in using the language.*

*All variables in a program reside in memory; the statements*

*float x;*

*x = 6.5;*

*request that the compiler reserve 4 bytes of memory (on a 32-bit computer) for the floating-point variable x, then put the ``value'' 6.5 in it.*

*Sometimes we want to know where a variable resides in memory. The address (location in memory) of any variable is obtained by placing the operator ``&'' before its name. Therefore &ampx is the address of x. C allows us to go one stage further and define a variable, called a pointer, that contains the address of (i.e. ``points to'') other variables. For example:*

*float x;*

*float px;*

*x = 6.5;*

*px = &x;*

*defines px to be a pointer to objects of type float, and sets it equal to the address of x:*

*Pointer use for a variable*

*The content of the memory location referenced by a pointer is obtained using the `` '' operator (this is called dereferencing the pointer). Thus, px refers to the value of x.*

*C allows us to perform arithmetic operations using pointers, but beware that the ``unit'' in pointer arithmetic is the size (in bytes) of the object to which the pointer points. For example, if px is a pointer to a variable x of type float, then the expression px + 1 refers not to the next bit or byte in memory but to the location of the next float after x (4 bytes away on most workstations); if x were of type double, then px + 1 would refer to a location 8 bytes (the size of a double)away, and so on. Only if x is of type char will px + 1 actually refer to the next byte in memory.*

*Thus, in*

*char pc;*

*float px;*

*float x;*

*x = 6.5;*

*px = &x;*

*pc = (char ) px;*

*(the (char ) in the last line is a ``cast'', which converts one data type to another), px and pc both point to the same location in memory--the address of x--but px + 1 and pc + 1 point to different memory locations.*

*Arrays :->*

*Arrays of any type can be formed in C. The syntax is simple:*

*type name[dim];*

*In C, arrays starts at position 0. The elements of the array occupy adjacent locations in memory. C treats the name of the array as if it were a pointer to the first element--this is important in understanding how to do arithmetic with arrays. Thus, if v is an array, v is the same thing as v[0], (v+1) is the same thing as v[1], and so on*

*8. Character Arrays*

*A string constant , such as*

*"I am a string"*

*is an array of characters. It is represented internally in C by the ASCII characters in the string, i.e., ``I'', blank, ``a'', ``m'',... for the above string, and terminated by the special null character ``\0'' so programs can find the end of the string.*

*String constants are often used in making the output of code intelligible using printf ;*

*printf("Hello, world\n");*

*printf("The value of a is: %f\n", a);*

*String constants can be associated with variables. C provides the char type variable, which can contain one character--1 byte--at a time. A character string is stored in an array of character type, one ASCII character per location. Never forget that, since strings are conventionally terminated by the null character ``\0'', we require one extra storage location in the array!*

*Functions :->*

*Functions are easy to use; they allow complicated programs to be parcelled up into small blocks, each of which is easier to write, read, and maintain. We have already encountered the function main and made use of I/O and mathematical routines from the standard libraries. Now let's look at some other library functions, and how to write and use our own.*

*Calling a Function*

*The call to a function in C simply entails referencing its name with the appropriate arguments. The C compiler checks for compatibility between the arguments in the calling sequence and the definition of the function.*

*Library functions are generally not available to us in source form. Argument type checking is accomplished through the use of header files (like stdio.h) which contain all the necessary information. For example, as we saw earlier, in order to use the standard mathematical library you must include math.h via the statement*

*#include < math.h>*

*at the top of the file containing your code. The most commonly used header files are*

*< stdio.h> -> defining I/O routines*

*< ctype.h> -> defining character manipulation routines*

*< string.h> -> defining string manipulation routines*

*< math.h> -> defining mathematical routines*

*< stdlib.h> -> defining number conversion, storage allocation*

*and similar tasks*

*< stdarg.h> -> defining libraries to handle routines with variable*

*numbers of arguments*

*< time.h> -> defining time-manipulation routines*

*In addition, the following header files exist:*

*< assert.h> -> defining diagnostic routines*

*< setjmp.h> -> defining non-local function calls*

*< signal.h> -> defining signal handlers*

*< limits.h> -> defining constants of the int type*

*< float.h> -> defining constants of the float type*

*Writing Your Own Functions*

*A function has the following layout:*

*return-type function-name ( argument-list-if-necessary )*

*{*

*...local-declarations...*

*...statements...*

*return return-value;*

*}*

*If return-type is omitted, C defaults to int. The return-value must be of the declared type.*

*A function may simply perform a task without returning any value, in which case it has the following layout:*

*void function-name ( argument-list-if-necessary )*

*{*

*...local-declarations...*

*...statements...*

*}*

*Command-line arguments*

*It is standard practice in UNIX for information to be passed from the command line directly into a program through the use of one or more command-line arguments, or switches. Switches are typically used to modify the behavior of a program, or to set the values of some internal parameters. You have already encountered several of these--for example, the "ls" command lists the files in your current directory, but when the switch -l is added, "ls -l" produces a so-called ``long'' listing instead. Similarly, "ls -l -a" produces a long listing, including ``hidden'' files, the command "tail -20" prints out the last 20 lines of a file (instead of the default 10), and so on.*

*Conceptually, switches behave very much like arguments to functions within C, and they are passed to a C program from the operating system in precisely the same way as arguments are passed between functions. Up to now, the main() statements in our programs have had nothing between the parentheses. However, UNIX actually makes available to the program (whether the programmer chooses to use the information or not) two arguments to main: an array of character strings, conventionally called argv, and an integer, usually called argc, which specifies the number of strings in that array. The full statement of the first line of the program is*

*main(int argc, char argv)*

*(The syntax char argv declares argv to be a pointer to a pointer to a character, that is, a pointer to a character array (a character string)--in other words, an array of character strings. You could also write this as char argv[]. Don't worry too much about the details of the syntax, however--the use of the array will be made clearer below.)*

*When you run a program, the array argv contains, in order, all the information on the command line when you entered the command (strings are delineated by whitespace), including the command itself. The integer argc gives the total number of strings, and is therefore equal to equal to the number of arguments plus one*

*CONTROL STATEMENTS*

*Branching :->*

*The C language programs presented until now follows a sequential form of execution of statements. Many times it is required to alter the flow of the sequence of instructions. C language provides statements that can alter the flow of a sequence of instructions. These statements are called control statements. These statements help to jump from one part of the program to another. The control transfer may be conditional or unconditional.*

*if Statement:*

*The simplest form of the control statement is the If statement. It is very frequently used in decision making and allowing the flow of program execution.*

*The If structure has the following syntax*

*if (condition)  
statement;*

*The statement is any valid C’ language statement and the condition is any valid C’ language expression, frequently logical operators are used in the condition statement. The condition part should not end with a semicolon, since the condition and statement should be put together as a single statement. The command says if the condition is true then perform the following statement or If the condition is fake the computer skips the statement and moves on to the next instruction in the program.*

***If else construct:***

*The syntax of the If else construct is as follows:-*

*The if else is actually just on extension of the general format of if statement. If the result of the condition is true, then program statement 1 is executed, otherwise program statement 2 will be executed. If any case either program statement 1 is executed or program statement 2 is executed but not both when writing programs this else statement is so frequently required that almost all programming languages provide a special construct to handle this situation.*

*Syntax*

|  |
| --- |
| *a> if (condition1 && condition2 && condition3)  b> if (condition1 // condition2 // condition3)* |

*The syntax in the statement ‘a’ represents a complex if statement which combines different conditions using the and operator in this case if all the conditions are true only then the whole statement is considered to be true. Even if one condition is false the whole if statement is considered to be false.   
  
The statement ‘b’ uses the logical operator or (//) to group different expression to be checked. In this case if any one of the expression if found to be true the whole expression considered to be true, we can also uses the mixed expressions using logical operators and and or together.*

*Nested if Statement*

*The if statement may itself contain another if statement is known as nested if statement.*

*Syntax:*

|  |
| --- |
| *if (condition1)    if (condition2)       statement-1;    else       statement-2;    else        statement-3;* |

*The if statement may be nested as deeply as you need to nest it. One block of code will only be executed if two conditions are true. Condition 1 is tested first and then condition 2 is tested. The second if condition is nested in the first. The second if condition is tested only when the first condition is true else the program flow will skip to the corresponding else statement.*

*Loops :->*

*Most real programs contain some construct that loops within the program, performing repetitive actions on a stream of data or a region of memory. There are several ways to loop in C. Two of the most common are the while loop:*

*while (expression)*

*{*

*...block of statements to execute...*

*}*

*and the for loop:*

*for (expression\_1; expression\_2; expression\_3)*

*{*

*...block of statements to execute...*

*}*

*The while loop continues to loop until the conditional expression becomes false. The condition is tested upon entering the loop. Any logical construction (see below for a list) can be used in this context.*

*The for loop is a special case, and is equivalent to the following while loop:*

*expression\_1;*

*while (expression\_2)*

*{*

*...block of statements...*

*expression\_3;*

*}*

*For instance, the following structure is often encountered:*

*i = initial\_i;*

*while (i <= i\_max)*

*{*

*...block of statements...*

*i = i + i\_increment;*

*}*

*This structure may be rewritten in the easier syntax of the for loop as:*

*for (i = initial\_i; i <= i\_max; i = i + i\_increment)*

*{*

*...block of statements...*

*}*

*Infinite loops are possible (e.g. for(;;)), but not too good for your computer budget! C permits you to write an infinite loop, and provides the break statement to ``breakout '' of the loop. For example, consider the following (admittedly not-so-clean) re-write of the previous loop:*

*angle\_degree = 0;*

*for ( ; ; )*

*{*

*...block of statements...*

*angle\_degree = angle\_degree + 10;*

*if (angle\_degree == 360) break;*

*}*

*Jumping Statement :->*

*Jump Statements (C# Reference)*

*Branching is performed using jump statements, which cause an immediate transfer of the program control. The following keywords are used in jump statements:*

* *break*
* *continue*
* *goto*
* *return*
* *throw*

*Modularity*

*In the making of this Software I used basically modularity or functionality in coding part of this Software. I used so many functions in making of this Software and performing different types of task, which is easy to me, do my work with great efficiency. In modularity of this Software C++ supports very well. I divide the different task in different modules or function, where every module is performing its individual task and give the required results.*

*jj*

*C++ - Access Level:*

*The classes in C++ have 3 important access levels. They are Private, Public and Protected. The explanations are as follows.*

*Private:*

*The members are accessible only by the member functions or friend functions.*

*Protected:*

*These members are accessible by the member functions of the class and the classes which are derived from this class.*

*Public:*

*Accessible by any external member. Look at the sample class below.*

*C++ Tutorial - Example of a class:*

*class Example\_class //Sample Class for the C++ Tutorial   
   {  
       private:  
         int x; //Data member   
         int y; // Data member   
       public:   
         Example\_Class() //Constructor for the C++ tutorial   
         {   
             x = 0;  
             y = 0;  
         }  
       ~Example\_Class() //destructor for the C++ Tutorial   
       { }   
      int Add()   
      {   
         return x+y;  
      }  
}*

*File Handling*

*Many real life problems handle large volumes of data and in such situation; we need to use same devices such as floppy disk or hard disk to store the data. The data is stored in these devices using the concept of files. A file is a collection of related data stored in a particular area on the disk. Programs can be designed to perform the read and write operations on these files. The I/O system of C++ contains a set of classes that define the file handling methods. These include ifstream, ofstream and fstream. These classes are derived from fstreambase and the corresponding iostream class.*

*The details of file stream classes are.*

* *fstreambase*

*Provides operation common to the file stream. Serves as a base for fstream, ifstream and ofstream class. Contains open() and close() functions.*

* *ifstream*

*provides input operations. Contains open() with default input mode. Inherits the functions get(), getline(), read(), seekg() and tellg() functions from istream.*

* *ofstream*

*Provides output operations. Contains open() with default output mode. Inherits put(), seekg(), tellp() and write() functions from ostream.*

* *fstream*

*Provides support for simultaneous input and output operations. Contains open() with default input mode. Inherits all the functions from istream and ostream classes*

*H/W & S/W Requirement*

*HARDWARE REQUIRMENTS*

*The following table lists the minimum and recommended hardware requirements for deploying this Project.*

|  |  |  |
| --- | --- | --- |
| *Component* | *Minimum* | *Recommended* |
| *Processor* | *2.5 gigahertz (GHz)* | *Dual processors that are each 3 GHz or faster* |
| *RAM* | *1 gigabyte (GB)* | *2 GB* |
| *Disk* | *NTFS file system–formatted partition with a minimum of 3 GB of free space* | *NTFS file system–formatted partition with 3 GB of free space plus adequate free space for your Web sites* |
| *Drive* | *DVD drive* | *DVD drive or the source copied to a local or network-accessible drive* |
| *Display* | *1024 × 768* | *1024 × 768 or higher resolution monitor* |
| *Network* | *56 kilobits per second (Kbps) connection between client computers and server* | *56 Kbps or faster connection between client computers and server* |

*Software requirements*

*Turbo C++.*

***CODING***

*//// <<\*\*\*\*\*\*\*\*EMPLOYEE DATA BASE MANAGEMENT SYSTEM\*\*\*\*\*\*\*\*>>*

*#include<fstream.h>*

*#include<conio.h>*

*#include<stdio.h>*

*#include<iomanip.h>*

*#include<graphics.h>*

*#include<dos.h>*

*#include<string.h>*

*#include<stdlib.h>*

*#include<process.h>*

*//// <<\*\*\*\*\*\*\*\*Global Functions\*\*\*\*\*\*\*\*>>*

*void display();*

*void menu();*

*void exit();*

*void search\_des();*

*void search\_dep();*

*void old\_rec();*

*void readfile();*

*void deleter();*

*void names();*

*void screen1();*

*void date();*

*void password();*

*//// <<\*\*\*\*\*\*\*\*Start Of Personal Class\*\*\*\*\*\*\*\*>>*

*class personal*

*{*

*protected:*

*float age,ht,wt,n,chld;*

*long int id;*

*char fnme[20],mnme[20],lnme[20],ftnme[40],*

*hnme[40],ms,bg[10],sex,add[60],cty[20],state[20],*

*cnt[20],dte[20],lndno[30],mbno[20];*

*public:*

*void getdata(long int);*

*void showdata();*

*};*

*//// <<\*\*\*\*\*\*\*\*End Of The Personal Class\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*End Of Professional Class\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Showdata Of Personal Information\*\*\*\*\*\*\*\*>>*

*void personal::showdata()*

*{*

*cout<<"\t\t Personal Record"<<endl;*

*cout<<"\tEmployee's ID : " <<id<<endl;*

*cout<<"\tEmployee's Name : " <<fnme<<setw(5)<<mnme<<setw(5)<<lnme<<endl;*

*cout<<"\tFather's Name : " <<ftnme<<endl;*

*cout<<"\tDate Of Birth : " <<dte<<"\tAge : " <<age<<endl;*

*cout<<"\tSex : " <<sex<<"\t Marital Status : " <<ms<<endl;*

*if(sex=='F' && ms=='Y')*

*{*

*cout<<"\tHusband's Name : " <<hnme<<endl;*

*}*

*if(ms=='Y')*

*cout<<"\tNo. Of Children : " <<chld<<endl;*

*}*

*cout<<"\tAddress : " <<add<<","<<cty<<","<<","<<state<<","<<cnt<<endl;*

*cout<<"\tContact No.:-"<<" Landline No.:"<<lndno<<"\tMobile No.:" <<mbno<<endl;*

*cout<<"\tHeight(in cms) : " <<ht<<"\t Weight(in kg) : " <<wt<<endl;*

*cout<<"\tBlood Group : " <<bg<<endl;*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Showdata\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Indata Of Professional Information\*\*\*\*\*\*\*\*>>*

*void professional::indata()*

*{*

*clrscr();*

*cout<<endl;*

*cout<<"\t\t Professional Details "<<endl;*

*cout<<endl<<"\t Enter the Percentage gained in High School : ";*

*gets(mtrc);*

*cout<<endl<<"\t Enter the Percentage gained in Intermediate : ";*

*gets(itm);*

*cout<<endl<<"\t Mention the Bachelor's Degree : ";*

*gets(bd);*

*cout<<endl<<"\t Mention the Master's Degree(if any) : ";*

*gets(md);*

*cout<<endl<<"\t Any Other Course : " ;*

*gets(noc);*

*cout<<endl<<"\t Enter the Previous Experience(if any) : " ;*

*gets(pe);*

*cout<<endl<<"\t Enter the Date Of Joining : " ;*

*gets(doj);*

*cout<<endl<<"\t Designation : ";*

*gets(des);*

*cout<<endl<<"\t Department : ";*

*gets(dep);*

*cout<<endl<<"\t Enter the Basic Pay(in Rs.) : " ;*

*cin>>bp;*

*cout<<endl<<"\t Enter the Allowances(in Rs.) : " ;*

*cin>>al;*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Indata\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Completedata\*\*\*\*\*\*\*\*>>*

*void professional::completedata()*

*{*

*personal::showdata();*

*professional::outdata();*

*getch();*

*}*

*//// <<\*\*\*\*\*\*\*\*Writing Data From The File xyz.txt\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Main Screen Function\*\*\*\*\*\*\*\*>>*

*void screen1()*

*{*

*clrscr();*

*sleep(1);*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t \* \* \* "<<endl;*

*delay(200);*

*cout<<"\t \* \* "<<endl;*

*delay(200);*

*cout<<"\t \* \* "<<endl;*

*delay(200);*

*cout<<"\t \* \* "<<endl;*

*delay(200);*

*cout<<"\t \* \* "<<endl;*

*delay(200);*

*cout<<"\t\t \* \*"<<endl;*

*delay(200);*

*cout<<"\t\t \* \*"<<endl;*

*delay(500);*

*cout<<"\t\t\t WELCOME TO "<<endl<<endl<<endl;*

*delay(500);*

*cout<<"\t\t\t EMPLOYEE DATABASE MANAGEMENT "<<endl<<endl<<endl;*

*delay(500);*

*cout<<"\t\t\t SYSTEM ";*

*sleep(2);*

*clrscr();*

*date();*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Main Screen Function\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Date Function\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Password Function\*\*\*\*\*\*\*\*>>*

*void password()*

*{*

*clrscr();*

*int j=0;*

*int c;*

*char password[25],username[25];*

*char password1[25]="employee",username1[25]="employee";*

*gotoxy(20,7);*

*cout<<" Enter Username : ";*

*gets(username);*

*gotoxy(20,9);*

*cout<<" Enter Password : ";*

*int count =0;*

*while(1)*

*{*

*do*

*{*

*c=getch();*

*if(c=='\r')*

*{*

*password[j]=c;*

*break;*

*}*

*else*

*{*

*cout<<"\*";*

*password[j]=c;*

*j++;*

*}*

*}*

*if(n==s.id)*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\t RECORD FOUND "<<endl;*

*cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*flag=1;*

*s.completedata();*

*getch();*

*}*

*}*

*if(flag==1)*

*{*

*clrscr();*

*display();*

*}*

*else*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <RECORD NOT FOUND> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*getch();*

*clrscr();*

*display();*

*}*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Search By ID\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Search By Designation\*\*\*\*\*\*\*\*>>*

*void search\_des()*

*{*

*fstream f;*

*professional s;*

*char ds[30];*

*float flag=0;*

*clrscr();*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\t Enter The Designation Of The Employee : ";*

*gets(ds);*

*f.open("ss.dat",ios::in|ios::binary);*

*f.seekg(0);*

*while(f.read((char \*) &s,sizeof(s)))*

*{*

*if(s.check1(ds))*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\t RECORD FOUND "<<endl;*

*cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*flag=1;*

*s.completedata();*

*getch();*

*}*

*}*

*if(flag==1)*

*{*

*clrscr();*

*display();*

*}*

*else*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <Record Not Found> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*getch();*

*clrscr();*

*display();*

*}*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Search By Designation\*\*\*\*\*\*\*\*>>*

*if(flag==1)*

*{*

*clrscr();*

*goto upmenu;*

*}*

*else*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <Match Not Found> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*f.close();*

*getch();*

*clrscr();*

*goto upmenu;*

*}*

*break;*

*//// <<\*\*\*\*\*\*\*\*End Of Update Personal Record\*\*\*\*\*\*\*\*>>*

*case 7:*

*clrscr();*

*goto umenu;*

*break;*

*case 8:*

*clrscr();*

*menu();*

*break;*

*default:*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*cout<<"\t\t You Have Entered The Wrong Choice.Enter Again "<<endl;*

*cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*goto upmenu;*

*}*

*//// <<\*\*\*\*\*\*\*\*Update Professional Record\*\*\*\*\*\*\*\*>>*

*usmenu:*

*case 2:*

*clrscr();*

*cout<<endl<<endl;*

*cout<<"\t\t Update Professinal Record "<<endl;*

*cout<<"\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "<<endl;*

*cout<<"\t\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* "<<endl<<endl;*

*cout<<"\t 1). Update Designation"<<endl;*

*cout<<"\t 2). Update Department"<<endl;*

*cout<<"\t 3). Update Basic Pay"<<endl;*

*cout<<"\t 4). Update Allowances"<<endl;*

*cout<<"\t 5). Go To Update Menu"<<endl;*

*cout<<"\t 6). Go To Main Menu "<<endl<<endl;*

*cout<<"\t Enter Your Choice : ";*

*cin>>w;*

*switch(w)*

*{*

*case 1:*

*clrscr();*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\t Enter The ID No. Of The Employee : ";*

*cin>>a;*

*f.open("ss.dat",ios::in | ios::out | ios::binary | ios::ate);*

*f.seekg(0);*

*while(f.read((char \*) &s,sizeof(s)))*

*{*

*if(a==s.id)*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\t MATCH FOUND "<<endl;*

*cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*i=sizeof(s);*

*f.seekp(-i,ios::cur);*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\* Enter New Designation : ";*

*gets(newdes);*

*cout<<endl;*

*cout<<"\t\* Old Designation is : "<<s.des<<endl;*

*strcpy(s.des,newdes);*

*cout<<"\t\* Changed Designation is : "<<s.des<<endl;*

*f.write((char \*) &s,sizeof(s));*

*flag=1;*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*}*

*if(flag==1)*

*{*

*clrscr();*

*goto usmenu;*

*}*

*else*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <Match Not Found> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*f.close();*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*break;*

*case 2:*

*clrscr();*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\t Enter The ID No. Of The Employee : ";*

*cin>>a;*

*f.open("ss.dat",ios::in | ios::out | ios::binary | ios::ate);*

*f.seekg(0);*

*while(f.read((char \*) &s,sizeof(s)))*

*{*

*if(a==s.id)*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\t MATCH FOUND "<<endl;*

*cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*i=sizeof(s);*

*f.seekp(-i,ios::cur);*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\* Enter New Department : ";*

*gets(newdep);*

*cout<<endl;*

*cout<<"\t\* Old Department is : "<<s.dep<<endl;*

*strcpy(s.dep,newdep);*

*cout<<"\t\* Changed Department is : "<<s.dep<<endl;*

*f.write((char \*) &s,sizeof(s));*

*flag=1;*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*}*

*if(flag==1)*

*{*

*clrscr();*

*goto usmenu;*

*}*

*else*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <Match Not Found> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*f.close();*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*break;*

*case 3:*

*clrscr();*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\t Enter The ID No. Of The Employee : ";*

*cin>>a;*

*f.open("ss.dat",ios::in | ios::out | ios::binary | ios::ate);*

*f.seekg(0);*

*while(f.read((char \*) &s,sizeof(s)))*

*{*

*if(a==s.id)*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\t MATCH FOUND "<<endl;*

*cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*i=sizeof(s);*

*f.seekp(-i,ios::cur);*

*cout<<"\t\* Enter New Basic Pay : ";*

*cin>>newbp;*

*cout<<endl;*

*cout<<"\t\* Old Basic Pay is : "<<s.bp<<endl;*

*s.bp=newbp;*

*cout<<"\t\* Changed Basic Pay is : "<<s.bp<<endl;*

*f.write((char \*) &s,sizeof(s));*

*flag=1;*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*}*

*if(flag==1)*

*{*

*clrscr();*

*goto usmenu;*

*}*

*else*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <Match Not Found> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*f.close();*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*case 4:*

*clrscr();*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\t Enter The ID No. Of The Employee : ";*

*cin>>a;*

*f.open("ss.dat",ios::in | ios::out | ios::binary | ios::ate);*

*f.seekg(0);*

*while(f.read((char \*) &s,sizeof(s)))*

*{*

*if(a==s.id)*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\t MATCH FOUND "<<endl;*

*cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*i=sizeof(s);*

*f.seekp(-i,ios::cur);*

*cout<<"\t\* Enter New Allowances : ";*

*cin>>newal;*

*cout<<endl;*

*cout<<"\t\* Old Allowances is : "<<s.al<<endl;*

*s.al=newal;*

*cout<<"\t\* Changed Allowances is : "<<s.al<<endl;*

*f.write((char \*) &s,sizeof(s));*

*flag=1;*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*}*

*if(flag==1)*

*{*

*clrscr();*

*goto usmenu;*

*}*

*else*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <Match Not Found> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*f.close();*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Update Professional Record\*\*\*\*\*\*\*\*>>*

*case 5:*

*clrscr();*

*goto usmenu;*

*break;*

*case 6:*

*clrscr();*

*menu();*

*break;*

*default:*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*cout<<"\t\t You Have Entered The Wrong Choice.Enter Again "<<endl;*

*cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*goto usmenu;*

*}*

*case 3:*

*clrscr();*

*menu();*

*break;*

*default:*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*cout<<"\t\t You Have Entered The Wrong Choice.Enter Again "<<endl;*

*cout<<"\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*clrscr();*

*goto umenu;*

*}*

*}*

*//// <<\*\*\*\*\*\*\*\*Sub Function Of Main Menu- Delete Record\*\*\*\*\*\*\*\*>>*

*void deleter()*

*{*

*clrscr();*

*int a;*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*cout<<"\t You Are Ready To Delete A Record"<<endl;*

*cout<<"\t \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl<<endl<<endl;*

*cout<<"\t 1). Delete A Record By ID No."<<endl;*

*cout<<"\t 2). Go To Main Menu"<<endl<<endl;*

*cout<<"\t Enter Your Choice : ";*

*cin>>a;*

*switch(a)*

*{*

*case 1:*

*clrscr();*

*s.dfile();*

*break;*

*case 2:*

*clrscr();*

*menu();*

*break;*

*}*

*getch();*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Delete Record\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Delete Record Files\*\*\*\*\*\*\*\*>>*

*void professional::dfile()*

*{*

*fstream f1,f2,f3;*

*professional s;*

*long int n;*

*int flag=0;*

*f1.open("ss.dat",ios::in | ios::binary);*

*f2.open("ss2.dat",ios::out | ios::ate | ios::binary);*

*f3.open("ss3.dat",ios::out | ios::ate | ios::binary);*

*cout<<endl<<endl<<endl<<endl;*

*cout<<"\t\t Enter The ID No. Of The Employee : ";*

*cin>>n;*

*f1.seekg(0);*

*while(f1.read((char \*) &s,sizeof(s)))*

*{*

*if(s.id==n)*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t\t MATCH FOUND "<<endl;*

*cout<<"\t\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;*

*getch();*

*flag=1;*

*f2.write((char \*) &s,sizeof(s));*

*}*

*else*

*{*

*f3.write((char \*) &s,sizeof(s));*

*}*

*}*

*remove("ss.dat");*

*rename("ss2.dat","ss.dat");*

*clrscr();*

*deleter();*

*if(s.id!=n)*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| <Match Not Found> | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t| | "<<endl;*

*cout<<"\t\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*| "<<endl;*

*cout<<"\t\t--------------------------------------------- "<<endl;*

*getch();*

*clrscr();*

*deleter();*

*}*

*}*

*//// <<\*\*\*\*\*\*\*\*Sub Function Of Main Menu-Names\*\*\*\*\*\*\*\*>>*

*void names()*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<”Sunbmitted To:Er.Vivek Mahajan”<<endl;*

*cout<<" Under the Guidance of :- Er. Shivani Mahajan"<<endl<<endl;*

*getch();*

*clrscr();*

*menu();*

*}*

*//// <<\*\*\*\*\*\*\*\*Sub Function Of Main Menu-Exit\*\*\*\*\*\*\*\*>>*

*void exit()*

*{*

*clrscr();*

*cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;*

*cout<<"\t\t | | | | | | "<<endl;*

*cout<<"\t\t \*\*\*\*\*\* SIGNED OUT SUCCESSFULLY\*\*\*\*\*\* "<<endl;*

*cout<<"\t\t | | | | | | | | "<<endl;*

*sleep(1);*

*exit(0);*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of Exit\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*End Of Sub Functions Of Main Menu\*\*\*\*\*\*\*\*>>*

*//// <<\*\*\*\*\*\*\*\*Main Function\*\*\*\*\*\*\*\*>>*

*void main()*

*{*

*clrscr();*

*textcolor(10);*

*textbackground(0);*

*professional s;*

*screen1();*

*}*

*//// <<\*\*\*\*\*\*\*\*End Of The Main Function\*\*\*\*\*\*\*\*>>*

***SNAP SHOTS***



* *Welcome Screen*



* *Login Page*



* *Main Page*



* *Add Record*

*(Personal Information)*



*(Professional information)*



* *Display Menu*



* *Display All Record*



* *By ID NO.*



* *By Designation*



* *BY DEPARTMENT*



* *Update Menu*



* *Update Personal Record*



* *Update Professional Record*



* *Delete Record*



* *Delete By ID No.*



* *Exit*

***Security Measures & Maintenance***

Even after the proper analysis and design - once a system has been implemented - even then some security and maintenance measures are to be taken , so that system data can be avoided from theft, disclosure and intentional misuse. Following points should be kept in mind

1. The entry to the system is to be done thru proper channel like DBA or system manager.
2. Each user is assigned a time constraint, a user name and a password.
3. Each used should lass with different type of privileges.
4. Daily Virus scanning is to be done.
5. Daily backup on tapes, CDs, thumb drives & Zip disks is to be taken.
6. Data must be secure - when it is in the organization, in transportation and under communications.

**CONCLUSION**

No program or system design is perfect. Communication between the user and the designer is not always complete or clear, and time is usually short. This results in errors. The number and nature of errors in a new design depends on several factors:

* Communication between the user and the designer.
* Personal prejudice on the part of users in disclosing information.
* The programmer’s ability to generate code that reflects exactly the system specifications.
* The time frame for the design.

In the **“Employee database system”**, I have tried my best to cover successfully and accurately all the requirements.

***EPILOGUE***

As no system design is ever perfect as communication problem, programmers lack of knowledge or time constraints create errors. A perfect project is that in which there are minimum number of keypunching. The number and nature of errors in new design depends on several factors. Some of the responsible factors are: -

* Communication between the user and the designer
* The programmers ability to generate a code that reflects exactly the system Specifications
* The time frame for the design

In our Project **“Employee database system”** we have tried my best to cover each aspect carefully and accurately. On further analysis the drawbacks of this system may show themselves, which may further require modifications.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

***BIBLIOGRAPHY***

* **System Analysis And Designed By Awadh.**
* **Object Oriented Programming By Ebalaguswami .**
* **Object Oriented Programming By Vivek Mahajan**
* **Program In C++ By Sumita Arora.**
* **Software engg. By Jalote**